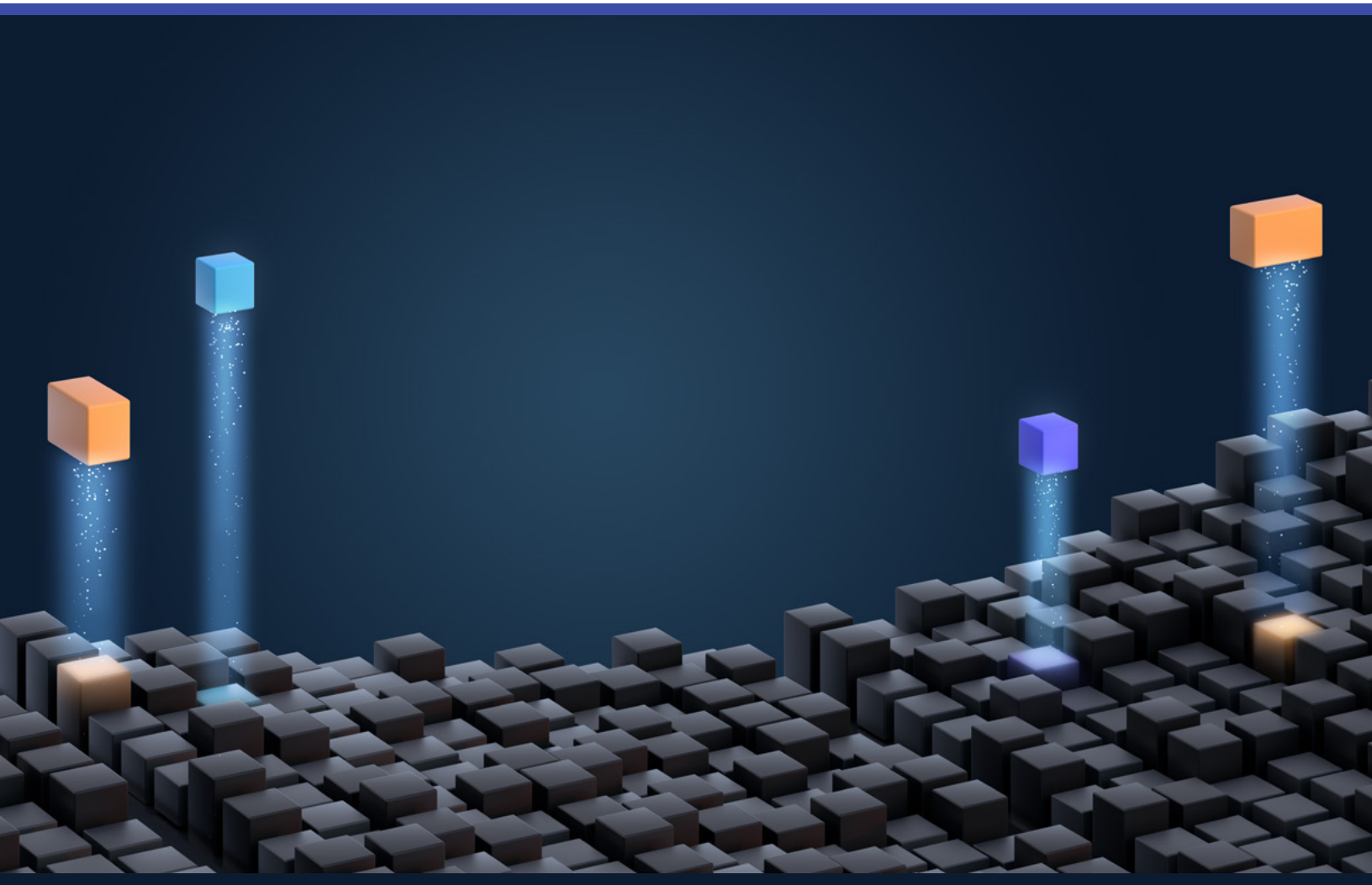
OpenLegacy
Digital to the Core

White Paper

Five ways to quickstart a digital transformation project for your legacy system

Avoid the cost, risk, time and resources normally associated with legacy modernization



www.openlegacy.com

Introduction

Most people think digital transformation is daunting, especially when you start with monolithic legacy systems, but it doesn't have to be. Overall the process for legacy integration shouldn't be any tougher than building an integration between two cloud systems. Here are five steps toward making legacy integrations easier, including comparisons between OpenLegacy and other approaches.

01

Get started quickly, even with limited resources

One common obstacle is getting the team resources approved. Everyone is busy, and managers may be hesitant to give up team members for a perceived big project. The good news is that the team can be small, which also aligns well with modern methodologies (Agile, Devops) that recommend using small cross-functional teams.

Most integration solutions require legacy team members to analyze and translate arcane data types and build facades on the legacy system for communication purposes, therefore these specialized team members are usually critical parts of a digital transformation project. However, OpenLegacy's platform removes the requirement for legacy team members by:

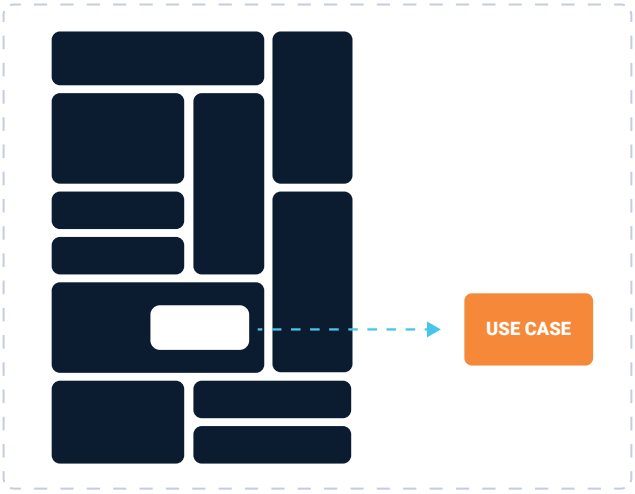
- Accessing legacy systems without making any changes
- Supporting the parsing of many different metadata types from screens to COBOL, RPG, and many other formats
- Generating APIs in a standard format (microservices) and language (Java), which allows teams to understand and use the APIs without specialized knowledge
- Generating a direct connection to the legacy system, which removes the requirement of a facade to translate the legacy data

Since rare, specialized legacy skills are not needed, you can simplify the process by using the same team composition as you use for cloud-only integration projects.

02

Start with a manageable use case

The current trend in development is to start with a use case, build an MVP (minimum viable product) and extend from there. However, when people think about digital transformation, they give man-hour estimates in years for large chunks of the project rather than estimates to build an MVP. Why? Companies feel the need to understand the legacy system by manually translating whole sections, making rapid progress and MVPs impossible.



Complex monolithic system where a piece of functionality is extracted to be used for a use case

Manual translation of entire systems isn't required anymore with automatic parsing and API generation.

Pick an easily definable issue or pain that matters to the organization. From there, you can decide on the development style:

Contract-first:

If you use contract-first, you start with a business need, then build a contract or set of contracts (APIs) representing that need. From there figure out what assets you require to build the necessary functionality to realize the APIs. This is a top-down development approach.

Contract-last:

This approach starts by looking at your systems to see who is using specific functionality. From there look at requests for additional functionality and decide to build based upon those usage patterns. It's a bottom up way to figure out needed functionality and the best way to build that functionality into an MVP.

OpenLegacy's platform handles both approaches. Simply choose the components of the monolith you wish to expose, then OpenLegacy translates those, and exposes functionality as APIs while connecting to the underlying code. This allows you to quickly and easily complete the system translations and you have deployment ready APIs.

03

Avoid concerns over building and running back-end connections

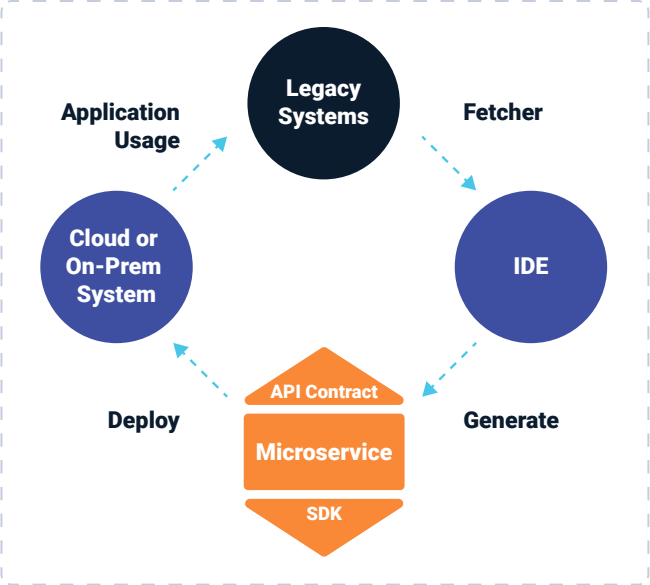
Many integrations stall out after a team decides on their APIs. Typically, teams have concerns about things like:

- Translating legacy parameters
- Mapping the parameters to the API
- Building in methods to pass the data
- Designing a systematic way of logging into the system
- Making sure the legacy system passes information efficiently

Furthermore, at design time companies often choose ESB/SOA systems with asynchronous queues to handle the passing of information, but then the team needs to build a facade on the legacy system for data translation. Designing the facade takes additional resources and makes the system more complex. This complexity not only slows down the project, but also facades don't fit well with modern development paradigms (DevOps). The whole process is time consuming and the results are less than desirable.

As mentioned, OpenLegacy's platform parses legacy parameters directly from the legacy system - therefore can bypass the ESB/SOA layers - and can generate Java code that handles the communication. At run-time the system is fast because it is a direct connection to the legacy system.

DevOps requires fast building and testing, but queues and facades makes users need to build and test the legacy platform every time a change occurs.

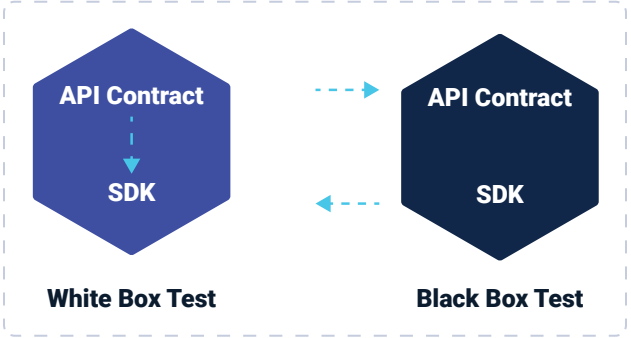


OpenLegacy's design-time and run-time processes for building and using microservice-based APIs for legacy systems

04

Automate testing—no one likes building tests

Most testing processes include both white-box and black-box testing. In the case of legacy integrations, the white-box approach tests the call from the API through to the legacy system. The black-box approach tests how the applications respond to data input on the API itself.



The white-box approach tests the call from the API through to the legacy system. The black-box approach tests how the applications respond to data input on the API itself.

Many companies have moved to tools like Jenkins to automate their testing process, but you still need to create the tests. Open source testing platforms have gained popularity over the years: JUnit is a popular choice for white box testing for Java and Swagger is the de-facto standard for black box testing of APIs.

OpenLegacy's platform handles the test creation and automates the testing process by generating:

- JUnit tests for the underlying code
- Swagger code for easy testing of APIs

The platform allows you to focus on design while still knowing the test is generated. If you change the API, the tests reflect that immediately.

OpenLegacy's platform simplifies the process by automating the generation of tests at the same time it generates the code for the APIs and calls to the legacy system.

05

Make deploying APIs simple and easy

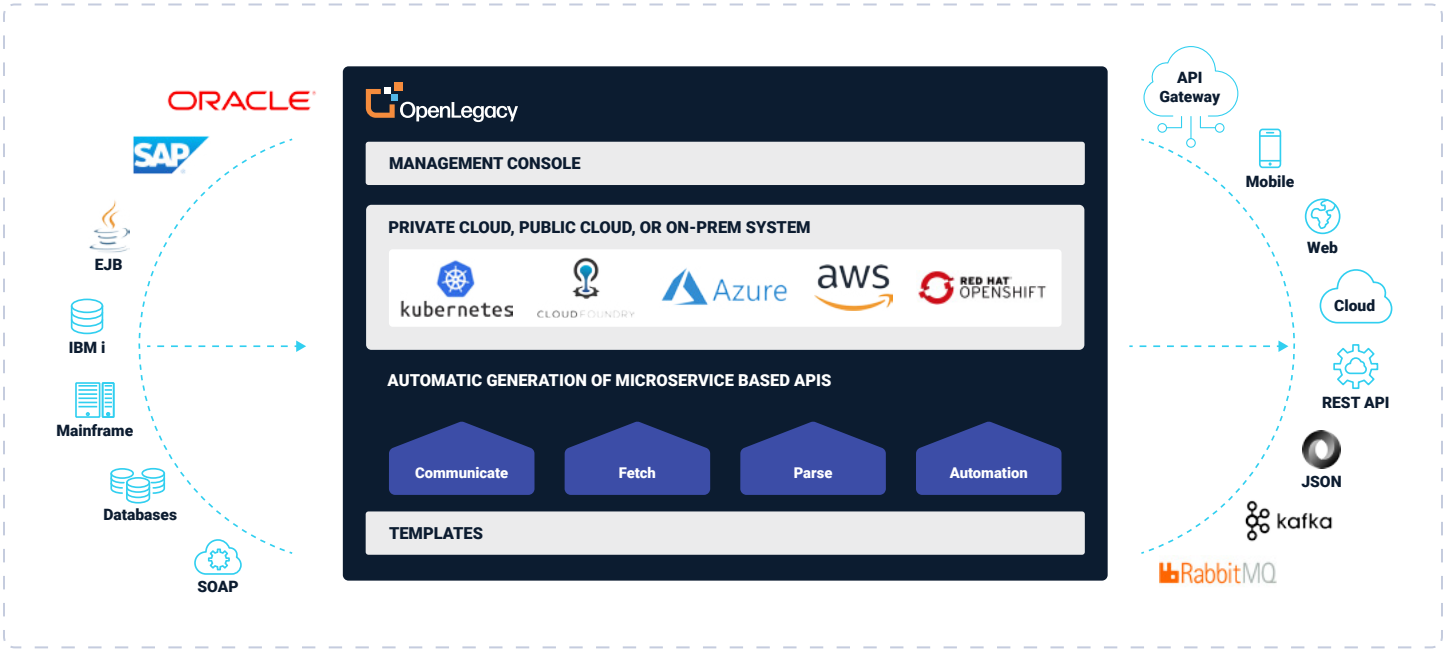
After building and testing the legacy system APIs, the question is how do you deploy the APIs? APIs by themselves are a contract with which applications have to comply in order to get data and complete business processes. In order to deploy an API, you also need the code, which fulfills the contract itself.

Microservices are a popular, flexible deployment method. A microservice is defined as a self-contained entity of limited scope. It also has a public interface and a representation of the business logic and data, while leaving the original system unchanged. The public interface is the API and the rest of the microservice is designed to fulfill the functionality of servicing the API.

For an API interfacing to a legacy system, the business logic converts the data between the API and legacy parameters. It also can include additional business logic for security and orchestration between multiple legacy data elements. Private methods inside the microservice contact the legacy system. This makes the system self-contained. Containers make microservices easy to deploy and maintain no matter where the services are needed.

OpenLegacy's platform generates the microservices including APIs and code to call the legacy system. The platform also includes Docker and configures any gateway to allow for easy loading into any system. Your company can choose to use microservices for the rest of the system or not. OpenLegacy's system still will work for you. The diagram below shows just some of the options for tools that can be integrated with OpenLegacy.

The platform simplifies the whole deployment process and allows you to focus on adding value to the data coming from the legacy system.



OpenLegacy platform's overall architecture showing its major functionality, the potential inputs, and how it outputs to a number of different types of systems.

Conclusion

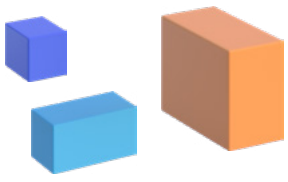
OpenLegacy's approach combines a highly automated platform with optional professional services as needed to help you quickly modernize your legacy applications, while starting with one project and one priority. We help you create and implement a digital transformation strategy that avoids the typical cost, risk and resources.

At no cost, you can see a demo of OpenLegacy using your code. You can also opt for a proof of concept, which typically takes a few days and not only creates some APIs, but also documents the results to help you build a business case.

Regardless of the reason you are looking to do a digital transformation, OpenLegacy can help you achieve it.

About OpenLegacy

OpenLegacy's Digital-Driven Integration enables organizations with legacy systems to release new digital services faster and more efficiently than ever before. It connects directly to even the most complex legacy systems, bypassing the need for extra layers of technology. It then automatically generates APIs in minutes, rapidly integrating those assets into exciting new innovations. Finally, it deploys them as standard microservices or serverless functions, giving organizations speed and flexibility while drastically cutting costs and resources. With OpenLegacy, industry-leading companies release new apps, features, and updates in days instead of months, enabling them to truly become digital to the core.



© OpenLegacy Inc. All Rights Reserved